# REDESIGN OF HIT/DIGIT/CLUSTER STRUCTURE AND CONTAINERS

# TPC Simulation Road Map

GAS SIMULATOR

HITS (dedx)

E Gen

ELECTRONS

Diff

Distort

PHITS

READOUT SIMULATOR

Amplif

Pad Resp

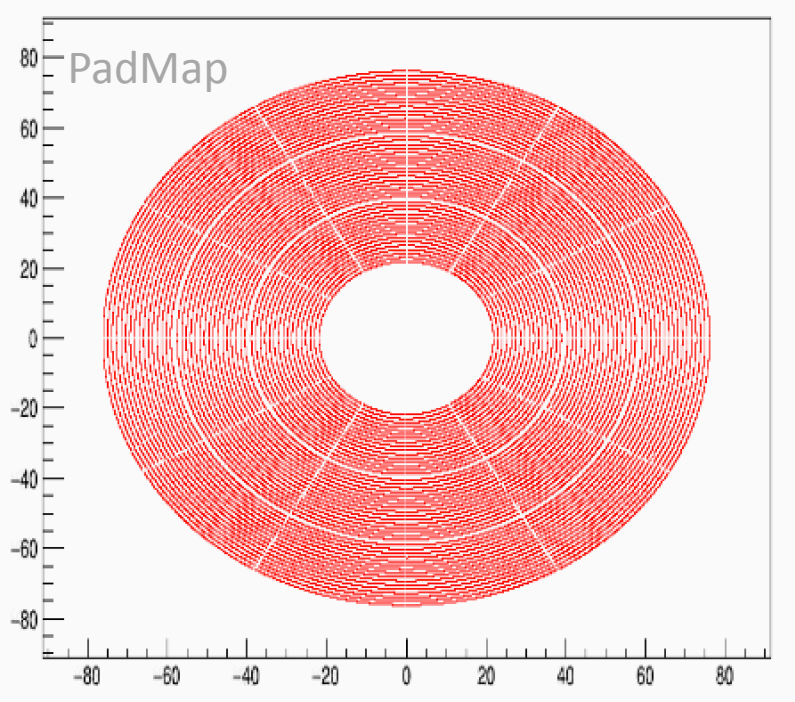
Pulse Gen

DIGITS

RAW READER

TRACKER

TRACKS



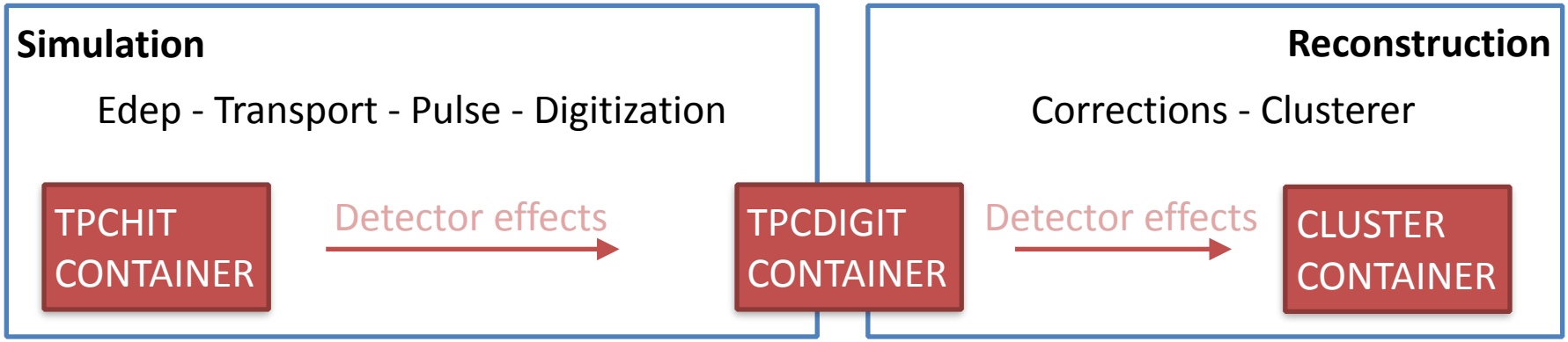

RECONSTRUCTOR

CLUSTERS

base/TPCParameters
base/TPCPadMap
simu/TPC_PHG4Detector
reco/TPCClusterer

presented in simulation meeting
https://indico.bnl.gov/conferenceDisplay.py?confId=2685

# Data Structure (TPC case)




TPCHITCONTAINER
vector<TPCHIT*> LIST[2]

TPCHIT : vHit(cylindrical)
Edep_Ion

vHit
x0, x1, x2, l, trkID

summary of G4Step

TPCDIGITCONTAINER
map<short,TPCDIGIT*> LIST[72]

TPCDIGIT : TPCCHANNEL
vector<float> TrainOfDigits

TPCCHANNEL : vChannel(PadMap)

vChannel
id1, id2

two ids allow for telescopic organisation

Cluster container not completed yet, but the structure will be a fusion of hit and digit concepts

# Advantages of New Data Structure

- Versatile vHit object: suitable hit coordinates (cartesian, cylindrical, hybrid)
- Data processing from hit to digit not constrained by container structure (e.g. layer) anymore
  - For TPC it means realistic distortions (dR,dPhi,dZ) in transport (spacecharge, ExB)
- Digits are stored in “binned” container -> local coordinates
  - Reduces time for cluster finder and pattern recognition
- Map object moves from local coordinates global coordinates (simultaneously in more than one coordinate system!)
- Way **faster** and **weightless** data handling than current structure: it **uses detector segmentation in data structure**

# List of Files [WIP]

```
[cperez@rcas2066 g4simulation]$ ls ROUTE/
vHit.cxx        vHit.h
vChannel.cxx    vChannel.h
```

```
[cperez@rcas2066 g4simulation]$ ls -R TPC/
TPC/:
base  reco  simu

TPC/base:
TPCChannel.h
TPCDigit.cxx                    TPCDigit.h
TPCCluster.cxx                  TPCCluster.h
TPCDigitContainer.cxx           TPCDigitContainer.h
TPCParameters.cxx               TPCParameters.h
TPCCorrections.cxx              TPCCorrections.h
TPCPadMap.cxx                   TPCPadMap.h
TPCPadMapCylindrical.cxx        TPCPadMapCylindrical.h
plotPadMapCylindrical.C

TPC/simu:
TPCHit.h
TPCCloud.h
TPCHitContainer.cxx                 TPCHitContainer.h
TPCDetectorSimulation.cxx           TPCDetectorSimulation.h
TPC_PHG4DetectorSubsystem.cxx       TPC_PHG4DetectorSubsystem.h
TPC_PHG4Detector.cxx                TPC_PHG4Detector.h
TPC_PHG4SteppingAction.cxx          TPC_PHG4SteppingAction.h

TPC/reco:
TPCClusterer.cxx                TPCClusterer.h
```

...

once fully tested, will commit

# ADDITIONAL MATERIAL

# List of Tasks Delimited in Last Tracking Meeting

sPhenix Tracking Tasks https://indico.bnl.gov/conferenceDisplay.py?confId=2964

## High Priority (Crucial for first implementation and MVTX Proposal)

Redesign hit and track structures - This is a basic building block of the project (Carlos, Haiwang)

- Minimize coordinate transformations, cache information
  - TPC coordinate map and cache (Carlos)
  - Silicon case (Tony)
- TVector operations?
- SvtxHit and SvtxTrack vs genfit::track and genfit::measurementOnPlane (Haiwang)
- Avoid duplication and parallel structures (Carlos)

Detector loop + hit containers - Efficient access and sorting of hits will determine performance

- Hits sorted by Layer
  - TPC hit/digit/cluster structure navigation (Carlos)
  - Silicon (MAPS + INTT) (Veronica)
  - Overlap treatment within Si layers (later)
- Direct access by Eta-Phi ranges
  - TPC coordinate map (Carlos)
  - Silicon (Veronica)
- Hit <-> detector plane association (generic container design for TPC + silicon, Haiwang + Carlos + Veronica)
- Alignment friendly implementation (keep in mind, hit needs to know which detector, Jin)
- Material budget per layer, active vs inactive detector components

GenFit - Key element to build trajectories (Haiwang)

- Turn Kalman Fitter into Filter for pattern recognition
- Isolate tools to calculate Chi2 increment for a given hit and TrajectoryState updates after adding a hit
- Provide easy to use getters
- Interface to material per layer (done, Jin)

TrajectoryBuilder class - Class to pull all elements together (Christof, Haiwang)

- Loop over seeds
- Track propagation
- Dynamic handling of track cloning and deletion
- Optimization of propagation strategy
  - Propagate each track to the end of the detector first vs propagating all tracks one layer at a time
  - Hit or track multiplicity may make caching more efficient in one case vs the other
- Track scoring (Sanghoon)
  - Decide if a trajectory needs to be kept or dropped based on holes in the track, chi2 etc.

Ambiguity resolution - Necessary to keep fake rate in check and to avoid duplication (Sanghoon)

- Check track overlaps based on shared hits
- Book keeping of hit usage. Unique hit <-> track association vs hit sharing?
- Releasing of hits from bad tracks

## Important (Needed for performance tuning of first implementation)

Definition of final Track Quality selection(Veronica + Sourav)

- Identify track quality criteria to protect against fakes while keeping the efficiency high
- Study impact of track quality on parameter estimation

Cluster validation -> Make optimal use of the detector information to estimate hit positions and errors

- Fix ITT hit position from simulation (Tony)
- TPC clustering, drift parameterization (Carlos)
- Hit sharing + clustering (Sourav)
- Cluster position determination. Parameterizations? (Sourav)

Performance evaluation (Sourav, Xiaolong)

- Efficiencies, fake rates, parameter resolutions, pull distributions, biases etc.
- Preparation of efficiency, fake rate correction tables
- Higher level checks, J/Psi mass peaks, HF/b-jet observables
- CPU performance

Optimization of hough tracking for seeding ( Sourav)

- Limiting PHG4HoughTransformTPC to the SI layers (0 - 7) works but gives shaky results

## Intermediate Term (After MVTX Proposal)

Vertexing (before tracking)

- Needed to limit combinatorics in seeding step
- Later as well

Track seeding

- Use Hough to get started
- If manpower available write modular seeding code (inside out vs outside in)

Tracking Iterations to optimize Efficiency

- Find easy to identify tracks first
- Remove hits from detector to reduce combinatorics
- Go for more difficult topologies

Electron resonstruction

- Gaussian Sum Filter extension to Kalman filter

## Long Term (Before Data Taking)

Repository Cleanup

- Split Reco code from Simulation – *Should be addressed after MVTX proposal*

Realistic Alignment functionality -> Analyze track residuals to position detector elements

- Database with positions and alignment of detector elements
- Alignment procedures based on physics data and/or cosmics
- Millipede algorithm?